We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

13

Multilevel Large-Scale Modules Floorplanning/Placement with Improved Neighborhood Exchange in Simulated Annealing

Kuan-Chung Wang[1] and Hung-Ming Chen[2]
[1]*SpringSoft, Inc., Hsinchu Science-Based Industrial Park*
[2]*Department of Electronics Engineering, National Chiao Tung University*
Hsinchu, Taiwan

1. Introduction

Modern system designs become more and more complex due to the progress of VLSI manufacturing technologies. In nanometer IC technologies and SoC (System on Chip) design flow, existing placement approaches face many serious challenges, including large size(billions of transistors), mix-size cell placement, wire congestion, and more complex design constraints (delay, noise, manufacturability, etc). Since the IC design market is more and more competitive, it is necessary to have faster time to market, smaller silicon area utilization, and less wire length for layout. Efficient and effective design methodologies of large scale design placement are essential for modern SoC designs.

Many placement methods have been presented in the literature (Chang et al., 2000; Guo et al., 1999; Lin & Chang, 2001; 2002a; Lin et al., 2003; Murata et al., 1995; Nakatake et al., 1996; Otten, 1982; Wong & Liu, 1986). Because of inflexibility in representing non-slicing placement and non-hierarchical data structures, the performance of traditional placement algorithms were not very good. Until recently, the B*-tree representation (Chang et al., 2000) provided an efficient, effective, and flexible data structure for non-slicing placement. Furthermore, MB*-tree algorithm (Lee et al., 2003) has presented multilevel framework which is more facilitating to solve large-scale floorplanning/placement problem. However applying simulated annealing approach in declustering stage spent much more time to search for better solutions.

On the other hand, the ϵ-neighborhood and λ-exchange algorithm, first presented in (Goto, 1981) and further applied in (Chan et al., 2000), was used for standard cell based placement. This method, for permuting cells with wire length driven approach, gave better performance compared with randomly interchanges of cells in simulated annealing paradigm. This limited trial permutation enable us to find a good local optimum solution more efficiently. The challenge lies in the modification of this approach to large-scale modules placement.

In this work, we transform the ϵ-neighborhood and λ-exchange to fit in the large-scale modules placement and use it in the refinement stage of MB*-tree algorithm. This method searches the solutions in the whole permutation of the selected modules. Although our ϵ-neighborhood and λ-exchange approach takes much time for one perturbation, its efficiency will compensate for the computation time by comparing with randomly interchanges, and more efficient in general compared with original MB*-tree. The results are encouraging. We have obtained

comparable or better results in area and wirelength metrics in less time spent (up to 30% improvement).
The remainder of this work is organized as follows. Section 2 gives a brief review on the B*-tree representation and MB*-tree, and describes previous ϵ-neighborhood and λ-exchange method. Section 3 presents our two-stage algorithm, clustering followed by declustering, mainly showing our effective refined approach to obtaining good candidates more efficiently. Section 4 shows the experimental results and Section 5 draws the conclusion.

2. Large-Scale Modules Placement with Neighborhood Exchange

In this section, we briefly review B*-tree representation and MB*-tree multilevel framework. We then introduce previous ϵ-neighborhood and λ-exchange method originally for standard cell placement.

2.1 Review of B*-tree and MB*-tree

Given a compacted placement that can neither move down nor move left called an *admissible placement*, we can represent it by a unique B*- tree (Chang et al., 2000) (See Figure 1 for the B*-tree representing the placement). A B*-tree is an ordered binary tree whose root corresponds to the module on the bottom-left corner. Using the depth-first search (DFS) procedure, the B*-tree for an admissible placement can be constructed in a recursive fashion. Starting from the root, we first recursively construct the left subtree and then the right subtree. Let R_i denote the set of modules located on the right-hand side and adjacent to m_i. The left child of the node n_i corresponds to the lowest module in R_i that is unvisited. The right child of n_i represents the lowest module located above m_i, with its x-coordinate equal to that of m_i. The B*-tree keeps the geometric relationship between two modules as follows. If node n_j is the left child of node n_i, module m_j must be located on the right-hand side of m_i, with $x_j = x_i + w_i$. Besides, if node n_j is the right child of n_i, module m_j must be located above module m_i, with the x-coordinate of m_j equal to that of m_i; i.e., $x_j = x_i$. Also, since the root of T represents the bottom-left module, the coordinate of the module is $(x_{root}, y_{root}) = (0, 0)$.
Inheriting from nice properties of ordered binary trees, the B*-tree is simple, efficient, effective, and flexible for handling non-slicing floorplans. It is particularly suitable for representing a non-slicing floorplan with various types of modules and for creating or incrementally updating a floorplan. What is more important, its binary-tree based structure directly corresponds to the framework of a hierarchical scheme, which makes it a superior data structure for multilevel large-scale building module floorplanning/placement. In (Lee et al., 2003), a multilevel floorplanning/placement framework based on the B*-tree representation, called MB*-tree, is presented to handle the floorplanning and packing for large-scale building modules. There were already many works that manipulated multilevel or hierarchical approach to disentangle the large scale issue in VLSI years ago: in graph/circuit partitioning such as Chaco (Hendrickson & Leland, 1995), hMetis (Karypis & Kumar, 1999), and ML (Alpert et al., 1998); in placement such as MPL (Chan et al., 2000); in routing such as MRS (Cong et al., 2001), MR (Lin & Chang, 2002b), and MARS (Cong et al., 2002).
The MB*-tree adopts a two-stage technique, clustering followed by declustering. The clustering stage iteratively groups a set of modules based on a cost metric guided by area utilization and module connectivity, and at the same time establishes the geometric relations for the newly clustered modules by constructing a corresponding B*-tree for them. The declustering stage iteratively ungroups a set of the previously clustered modules (i.e., perform tree expansion) and then refines the floorplanning/placement solution by using a simulated annealing

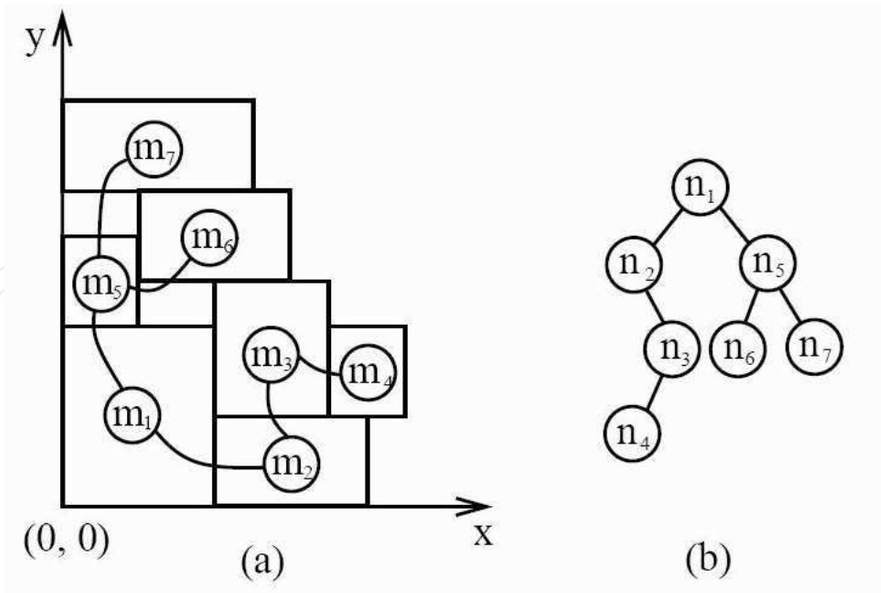


Fig. 1. An admissible placement and its corresponding B*-tree.

scheme. In particular, the MB*-tree preserves the geometric relations among modules during declustering, which makes the MB*-tree good for the multilevel floorplanning/placement framework.

2.2 Module Perturbation Based on Neighborhood Exchange

Those approaches were first brought forth in (Goto, 1981), and promoted in (Chan et al., 2000). But they are all about gate array/cell based placement. We first review those approaches in this subsection, then later show our improvement in our framework for efficient large-scale modules placement. Based on different definitions on ϵ-neighborhood and λ-exchange, we categorize them into two forms: unidirectional circulation form (UCF) and detoured circulation form (DCF).

2.2.1 Unidirectional Circulation Form

Consider a board on which every module is placed. Pick one module and move it while the other modules remain fixed. The wirelength of a signal net does not change as long as the signal net is not connected to this module. The median of module M is defined as a position where the routing length associated with module M is minimum. Then we sort all the wirelengths associated with module M with respect to the module M position in ascending order. Choose ϵ elements from the minimum one, the set of these ϵ positions is defined as the *ϵ-neighborhood for median of module M*.

Let S be the set of all feasible solutions of this placement and let x be a feasible solution, $x \in S$. Consider the neighborhood of x, denoted by $X(x)$, which is a subset of S. In the first step, x is set to a feasible solution and a search is made in $X(x)$ for a better solution x' to replace x. This process, which is referred to hereafter as a local transformation, is repeated until no such x' can be found. A solution x'' is said to be a local optimum if x'' is better than any other elements of $X(x)$. Many definitions may be considered for the neighborhood of a solution. The set of solutions transformable from x by exchanging not more than λ elements

is regarded as the neighboorhood of x. A solution x is said to be λ-optimum if x is better than any other solutions in the neighborhood in this sense. Although the λ-optimum solution gets better as λ increases, the computation time can easily go beyond the acceptable limit when an exhaustive search is performed for large λ. Therefore (Goto, 1981) presented the following method which does not examine all the elements in the neighborhood. The illustrations of this approach are shown in Figure 2 and 3. Figure 2 shows the corresponding search tree for module interchange and Figure 3 shows the trial interchange of modules. In this example, we set ϵ=3 and λ = 4.

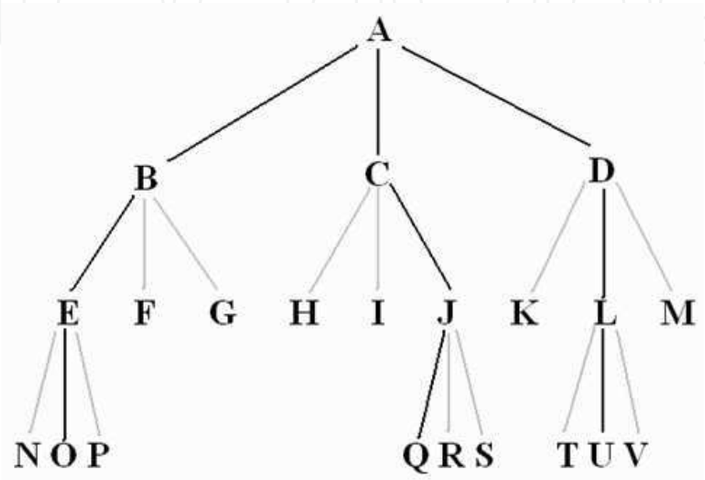


Fig. 2. The search tree of unidirectional circulation form, where ϵ=3 and λ = 4. Each node represents a module and each edge represents a trial transformation. A path connecting node A and one of the other nodes defines a possible interchange. The path $A \rightarrow B \rightarrow E \rightarrow O$ refers to the trial interchange of four modules, as shown in Figure 3. Module A is placed on the slot of B, then the median of B and its ϵ-neighborhood are generated. Here the ϵ-neighborhood module are E, F, and G. Thus interchanges $A \rightarrow B \rightarrow E$, $A \rightarrow B \rightarrow F$, and $A \rightarrow B \rightarrow G$ are tried.

2.2.2 Detoured Circulation Form

This form is presented in (Chan et al., 2000) and modified from previous form. Assuming all modules except module v are fixed in their current locations, we can compute v's optimal slot locations. Suppose v's optimal slot location is *(r,c)* where r is the row index and c is column index in our grid. Modules located in slots at *(i,j)*, where $|i\text{-}r|+|j\text{-}c| \leqq \epsilon$, are called ϵ-neighbors of module v (Figure 4).

λ-exchange algorithm used in (Chan et al., 2000) is different from UCF as well. Since the search tree from previous form grows rapidly with slight increase in ϵ and λ, and module exchange sequence may not be the best possible, λ-exchange procedure has been modified. Suppose v_1 is the first module to be moved. We find its ϵ-neighbors and randomly pick one module, say v_2, among these modules. Then for v_2, we find its ϵ-neighbors, and randomly pick one module, and continue in this fashion until we have λ modules. For the λ modules, we try all of their placement permutations (the total number is λ!) and exchange modules according to the least cost permutation. Figure 5 illustrates this change. Experimental results in (Chan et al., 2000) show that UCF algorithm quickly gets stuck in local minimum. Later we

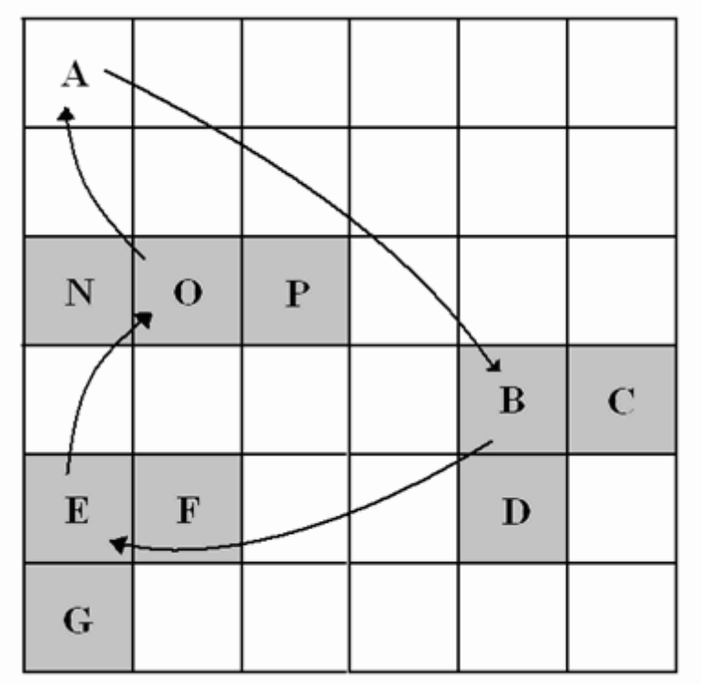


Fig. 3. Trial interchange of modules, $A \rightarrow B \rightarrow E \rightarrow O \rightarrow A$ in unidirectional circulation form. Module A is placed on the slot of B, B is placed on E, E on O, and O on A, in a round robin sequence. Although this transformation is a quadruple interchange, it includes a pairwise interchange as a special case, i.e., paths $A \rightarrow B$, $A \rightarrow C$, and $A \rightarrow D$.

show that we use detoured circulation form to develop our refined neighborhood exchange approach.

2.3 Problem Formulation

The problem we concerned about is described as follows, same as in MB*-tree. Let $M = \{m_1, m_2, ..., m_n\}$ be a set of n rectangular modules. Each module $m_i \in M$ is associated with a two tuple (h_i, w_i), where h_i and w_i denote the width and height of m_i, respectively. Let $N = \{n_1, n_2, ..., n_k\}$ be a set of k net. Each net $n_i \in N$ is a set of modules which are connected together. A placement $P = \{(x_i, y_i) \mid m_i \in M\}$ is an assignment of rectangular modules m_i's with the coordinates of their bottomleft corners being assigned to (x_i, y_i)'s so that no two modules overlap. The objective is to minimize a cost of combination of the area and half-perimeter wirelength.

3. The MBNE Algorithm

In this work, we decide to keep the multilevel hierarchy and the B*-tree representation of MB*-tree, but replace its simulated annealing refinement method by ϵ-neighborhood and λ-exchange algorithm for better performance. Since this algorithm combines the MB*-tree and ϵ-neighborhood and λ-exchange methods, we called it MBNE algorithm. We present our MBNE algorithm for multilevel large-scale building modules floorplanning/placement in this section. This algorithm adopts a two-stage approach, clustering followed by declustering, by using the B*-tree representation. Figure 6 shows the MBNE algorithm flow.

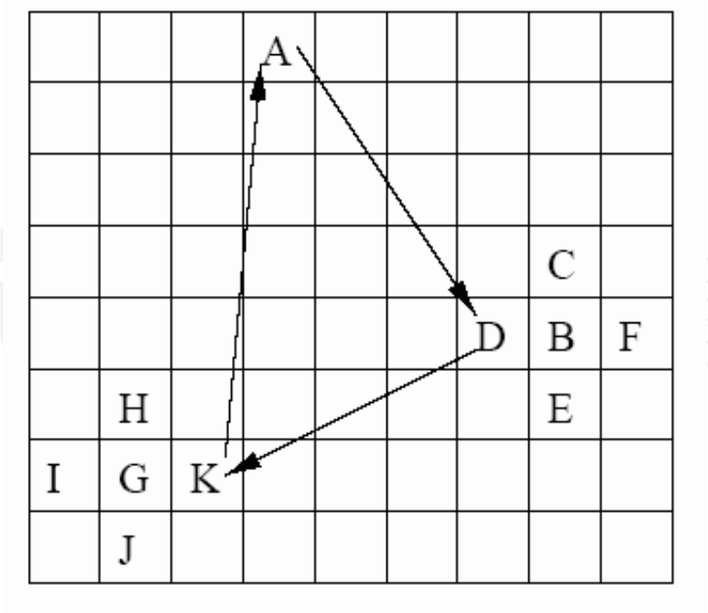


Fig. 4. The ϵ-neighbors in detoured circulation form. Suppose the optimal slot location of module A is occupied by module B. So A's 1-neighbors (ϵ = 1) are {B, C, D, E, F}. Similarly, assuming that D's optimal slot is taken by G, we define module D's 1-neighbors are {G, H, I, J, K}.

The clustering operation results in two types of modules, namely primitive modules and cluster modules. A primitive module m is a module given as an input (i.e., $m \in M$) while a cluster one is created by grouping two or more primitive modules. Each cluster module is created by a clustering scheme $\{m_i, m_j\}$, where m_i (m_j) denotes a primitive or a cluster module.
In the following subsections, we give a detailed review on clustering and declustering algorithms in MB*-tree (Lee et al., 2003) and our refinement approaches in declustering phase to improve the packing results.

3.1 The Clustering Phase

In this stage, we iteratively group a set of (primitive or cluster) modules until a single cluster is formed (or until the number of cluster modules is smaller than a threshold) based on a cost metric of area and connectivity. The clustering metric is defined by the two criteria: area utilization (dead space) and the connectivity density among modules.
The area utilization for clustering two modules m_i and m_j can be measured by the resulting dead space s_{ij}, representing the unused area after clustering m_i and m_j. Let s_{tot} denote the dead space in the final floorplan P. We have $s_{tot} = A_{tot} - \sum_{m_i \in M} A_i$, where A_i denotes the area of module m_i and A_{tot} the area of the final enclosing rectangle of P. Since $\sum_{m_i \in M} A_i$ is a constant, minimizing A_{tot} is equivalent to minimizing the dead space s_{tot}.
Let the connectivity c_{ij} denote the number of nets between two modules m_i and m_j . The connectivity density d_{ij} between two (primitive or cluster) modules m_i and m_j is given by

$$d_{ij} = c_{ij}/(n_i + n_j) \tag{1}$$

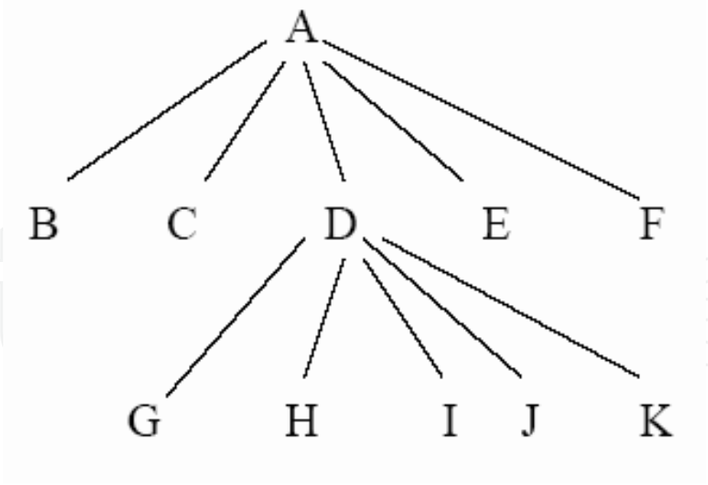


Fig. 5. Search tree from A in detoured circulation form. Suppose we pick modules A, D, and K. All six permutations will be tried: no exchange, $A \leftrightarrow D$, $A \leftrightarrow K$, $D \leftrightarrow K$, $A \rightarrow D \rightarrow K \rightarrow A$, $A \rightarrow K \rightarrow D \rightarrow A$.

where n_i (n_j) denotes the number of primitive modules in m_i (m_j). Often a bigger cluster implies a larger number of connections. The connectivity density considers not only the connectivity but also the sizes of clusters between two modules to avoid possible biases.
Obviously, the cost function of dead space is for area optimization while that of connectivity density is for timing and wiring area optimization. Therefore, the metric for clustering two (primitive or cluster) modules m_i and m_j, $\phi : \{m_i, m_j\} \rightarrow \Re^+ \bigcup \{0\}$, is then given by

$$\phi(\{m_i, m_j\}) = \alpha \hat{s}_{ij} + \frac{\beta}{\hat{d}_{ij}} \tag{2}$$

where $\hat{s}_{ij}$ and $\hat{d}_{ij}$ are respective normalized costs for s_{ij} and d_{ij}, α, β and K are user-specified parameters/constants.
Based on ϕ, we cluster a set of modules into one at each iteration by applying the aforementioned methods until a single cluster containing all primitive modules is formed or the number of modules is smaller than a given threshold. During clustering, we record how two modules m_i and m_j are clustered into a new cluster module m_k. If m_i is placed left to (below) m_j , then m_i is horizontally (vertically) related to m_j, n_j is the left (right) child of n_i in its corresponding B*-tree (see Figure 7). The relation for each pair of modules in a cluster is established and recorded in the corresponding B*-subtree during clustering. It will be used for determining how to expand a node into a corresponding B*-subtree during declustering.

3.2 The Declustering Phase

The declustering metric is defined by the two criteria: area utilization (dead space) and the wirelength among modules. Dead space is the same as that defined in previous subsection. The wirelength of a net is measured by half the bounding box of all the pins of the net, or by the length of the center-to-center interconnections between the modules if no pin positions are specified. The wirelength for clustering two modules m_i and m_j , w_{ij}, is measured by the total wirelength interconnecting the two modules. The total wirelength in the final floorplan P, w_{tot}, is the summation of the length of the wires interconnecting all modules.

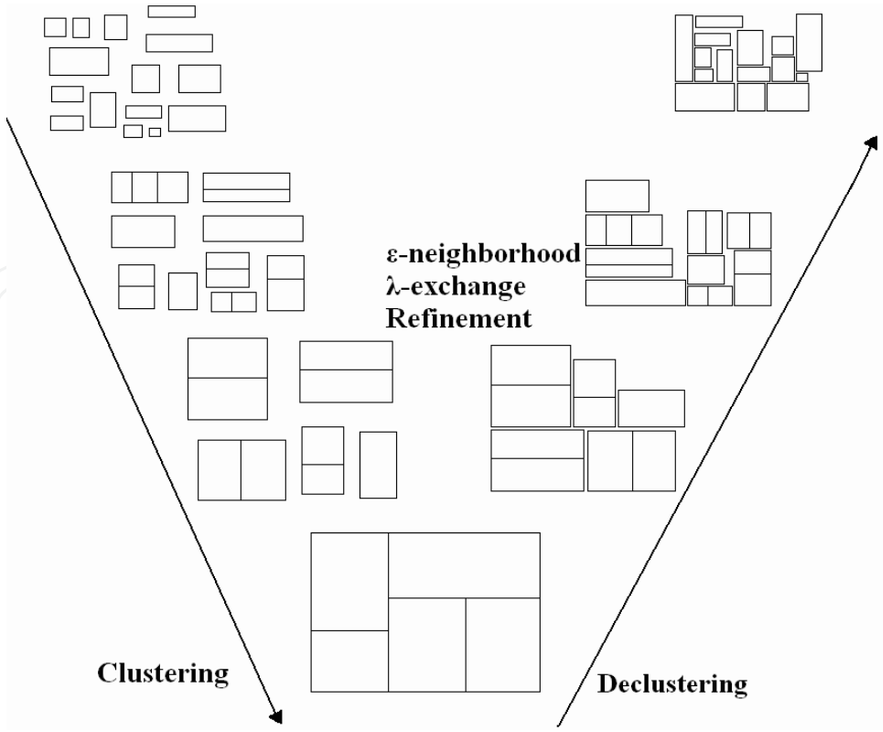


Fig. 6. The MBNE algorithm flow. First clustering, then followed by declustering, using our refined approaches to improve efficiency and the packing results.

Obviously, the cost function of dead space is for area optimization while that of wirelength is for timing and wiring area optimization. Therefore, the metric for refining a floorplan solution during declustering, ψ_{ij}:$\{m_i,m_j\}\rightarrow\Re^+\bigcup\{0\}$, is then given by

$$\psi_{ij} = \gamma\hat{s}_{ij} + \delta\hat{w}_{ij} \tag{3}$$

where $\hat{s}_{ij}$ and $\hat{w}_{ij}$ are respective normalized costs for s_{ij} and w_{ij}, and γ and δ are user-specified parameters.

In our approach, the declustering stage iteratively ungroups a set of previously clustered modules (i.e., expand a node into a subtree according to the B*-tree constructed at the clustering stage) and then refines the floorplan solution based on the ϵ-neighborhood and λ-exchange method. We apply the same declustering algorithm shown in (Lee et al., 2003) in our MBNE algorithm.

3.3 Our Refined Neighborhood Exchange Approach

At all levels of declustering, we apply the ϵ-neighborhood and λ-exchange method to refine the floorplan for gaining a better solution. We redefine the ϵ and λ in the B*-tree representation. The original definition of ϵ-neighborhood of module v in (Goto, 1981) is the modules located in slots at row i, column j where $|i\text{-}r|+|j\text{-}c|\leqq\epsilon$ and (r,c) is the optimal slot location of v. But in the non-slicing placement of large-scale circuit, the optimal slot location is hard to compute and it will shift when perturbing the modules. Hence we redefine the ϵ-neighborhood of module v as the modules away from v within ϵ branches in B*-tree. Figure 8 shows 1-neighborhood and 2-neighborhood of module n_2.

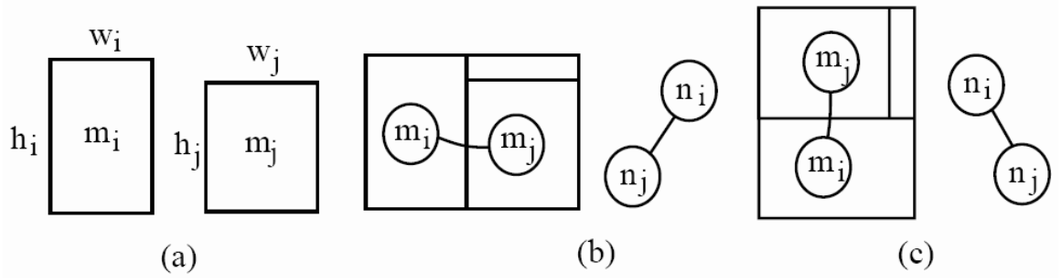


Fig. 7. The relation of two modules and their clustering (Lee et al., 2003) (a) Two candidate modules m_i and m_j. (b) The clustering and the corresponding B*-subtree for the case where m_i is horizontally related to m_j. (c) The clustering and the corresponding B*-subtree for the case where m_i is vertically related to m_j.

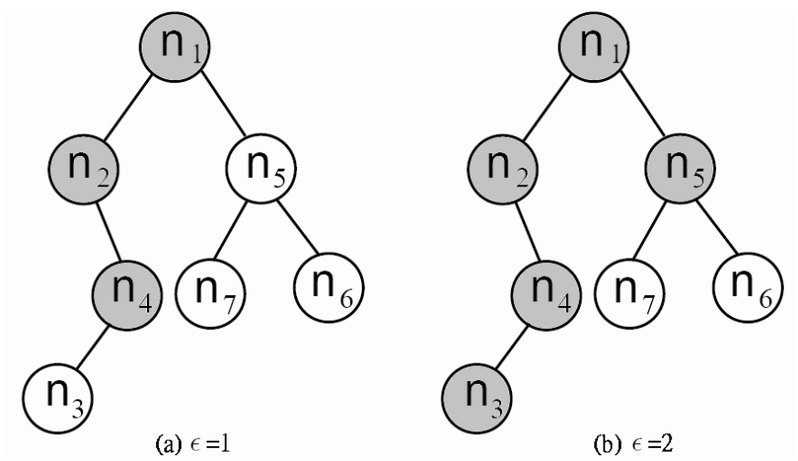


Fig. 8. The definition of ϵ-neighborhood in our refined neighborhood exchange approach. (a)ϵ=1 and (b)ϵ=2. The highlighted nodes besides node n_2 are the neighborhood modules.

In our refined neighborhood exchange algorithm, first we choose a starting module A, and select the module B which in the same net with module A. We then randomly pick the module B_λ in the ϵ-neighbors of module B, so we have A and B_λ for 2-exchange now. Furthermore, we can continue selecting the module C which in the same net with A and B, and randomly pick the module C_λ in ϵ-neighbors of module C for 3-exchange. Do this sequence until we have λ modules for λ-exchange (see Figure 9).
After we get all the λ modules, we try all of their placement permutations. Since this is a large-scale circuit placement, modules normally have different heights and widths. Therefore the rotation of modules will affect the placement's result. The total number of permutations is $\lambda! \times 2^\lambda$. Finally, we keep the permutation with the lowest cost and start the next turn of refinement.

3.4 Novel Move Based on Null Module Insertion

Our ϵ-neighborhood and λ-exchange approach can rotate and/or swap the modules to perturb the placement, but it can not move a module to another place. Thus, we replace one of the λ-exchange modules by null module for permutations. The null module does not connect to any module, and its height and width are equal to zero. When we decide to use null module

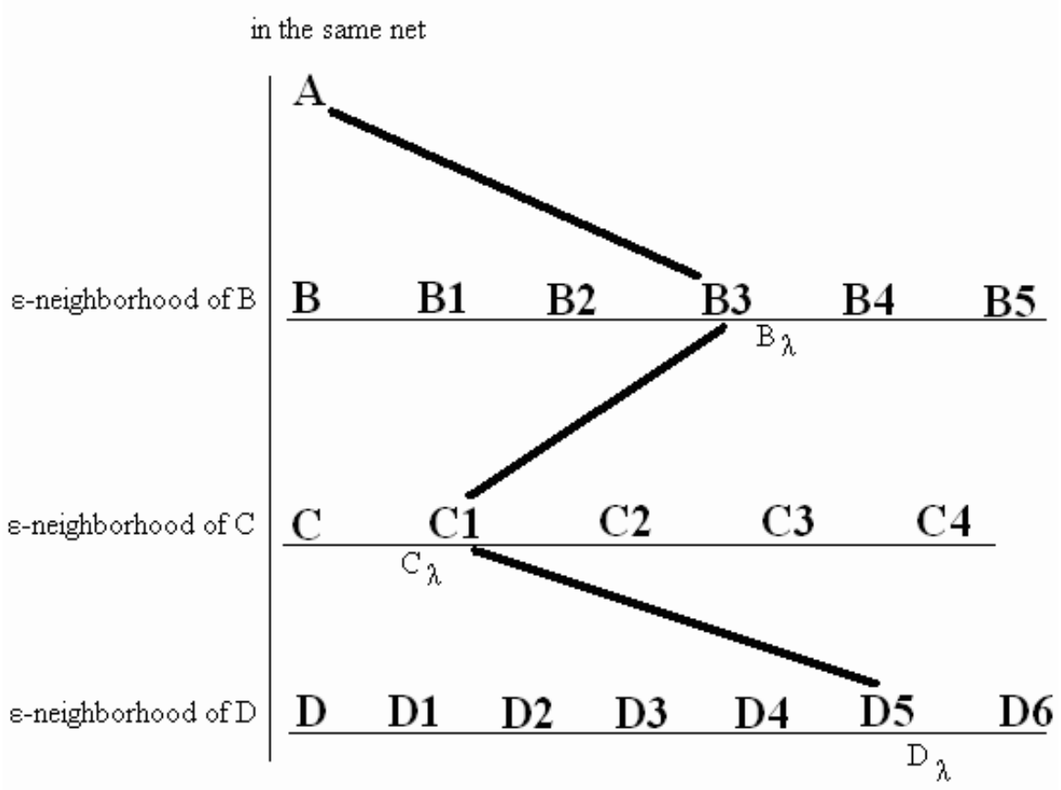


Fig. 9. An example of λ-exchange in our approach, where λ=4.

by some probability, we insert it to be the replaced λ-exchange module's child. When a module swap with the null module, it is equivalent with moving the module to be the replaced module's child. Figure 10 is an example of null module insertion for refinement.
We have applied the null module in the ϵ-neighborhood and λ-exchange refinement, so we can combine the original three operations (rotate, move, and swap) to perturb the placement, and get the lowest cost one.

3.5 Floorplanner Flow

The MBNE algorithm integrates the aforementioned three algorithms. We first perform clustering to reduce the problem size level by level and then enter the declustering stage. In the declustering stage, we perform floorplanning for the modules at each level using the ϵ-neighborhood and λ-exchange algorithm for refinement.
Figure 11 illustrates an execution of the MBNE algorithm (from (Lee et al., 2003)). For explanation, we cluster three modules each time. Figure 11(a) lists seven modules to be packed, m_i's, $1 \leq i \leq 7$. Figure 11(b)-(d) illustrates the execution of the clustering algorithm. Figure 11(b) shows the resulting configuration after clustering m_5, m_6, and m_7 into a new cluster module m_8 (i.e., the clustering scheme of m_8 is $\{\{m_5,m_6\},m_7\}$). Similarly, we cluster m_1,m_2, and m_4 into m_9 by using the clustering scheme $\{\{m_2, m_4\}, m_1\}$. Finally, we cluster m_3,m_8, and m_9 into m_{10} by using the clustering scheme $\{\{m_3, m_8\}, m_9\}$. The clustering stage is done, and the declustering stage begins, in which ϵ-neighborhood and λ-exchange method are applied to refine the coarse floorplan. In Figure 11(e), we first decluster m_{10} into m_3, m_8, and m_9 (i.e., expand the

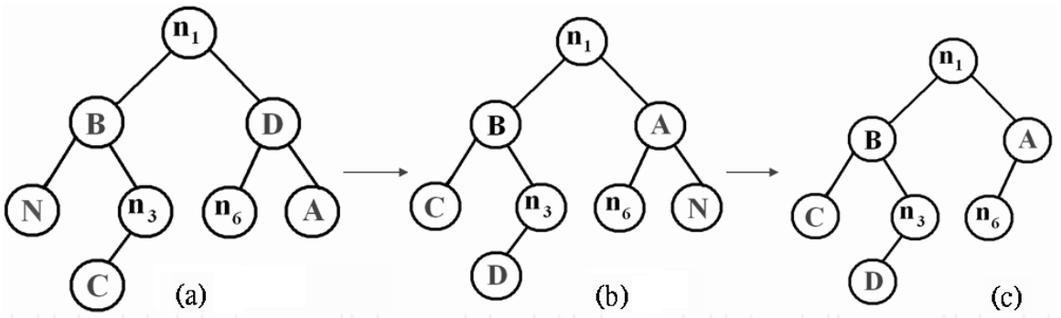


Fig. 10. An example of null module insertion in refined neighborhood exchange.(a)Insert modele N to replace modele B for swaping. (b)After swap $A \rightarrow D \rightarrow C \rightarrow N \rightarrow A$. (c)Delete module N.

node n_{10} into the B*-subtree illustrated in Figure 11(e)). We then move m_8 to the top of m_9 (perform Op2 for m_8) during ϵ-neighborhood and λ-exchange refinement (see Figure 11(f)). As shown in Figure 11(g), we further decluster m_9 into m_1, m_2, and m_4, and then rotate m_2 and move m_3 on top of m_2 (perform Op1 on m_2 and Op2 on m_3), resulting in the configuration shown in Figure 11(h). Finally, we decluster m_8 shown in Figure 11(i) to m_5, m_6, and m_7, and move m_4 to the right of m_3 (perform Op2 for m_4), which results in placement with good quality shown in Figure 11(j).

4. Experimental Results

We implement the MBNE algorithm in C++ programming language. The platform is Intel Pentium 4 2.4GHz CPU with 1.5GB memory. We have compared our approach with the MB*-tree algorithm on benchmarks including *industry* (Lee et al., 2003), MCNC and GSRC benchmarks for area, wirelength and simultaneous area and wirelength optimizations.

The circuit *industry* is a 0.18μm, 1GHz industrial design with 189 modules, 20 million gates and 9,777 center-to-center interconnections. It is a large chip design and consists of three modules with aspect ratios greater than 19 and as large as 36. Table 1 shows the results of MBNE compared with MB*-tree for this circuit. In each entry of the table, we list the best/average values obtained in ten runs of MBNE and MB*-tree. We have achieved less area and wirelength (WL) in averagely less time.

The *ami*49 is the largest MCNC benchmark circuit, (Lee et al., 2003) has created seven synthetic circuits, named ami49_x, by duplicating the modules of ami49 by x times to test the capability of our algorithm. The largest circuit ami49_200 contains 9800 modules. Moreover, we use GSRC benchmarks which contains n100, n200, and n300 circuits as our experimental suites. Table 2 and Table 3 shows the results of MBNE compared with MB*-tree in these two sets of benchmarks. Again we have acheived less area and wirelength in less runtime. The reason is that we use our refined neighborhood exchange approach to effectively and efficiently search for solution cadidates, instead of near-random simulated annealing.

For demostrating the efficiency, we choose four circuits from the *industry*, MCNC, and GSRC benchmark to compare for efficiency between MBNE and MB*-tree algorithm. We spend 70% of runtime compared with MB*-tree algorithm for four circuits. Table 4 shows the results of area, dead space and runtime of MBNE and MB*-tree. MBNE obtains dead space of 2.34%, 2.11%, 2.89% and 2.32% while MB*-tree requires dead space of 2.32%, 2.62%, 3.18% and 3.84% in these four circuits.

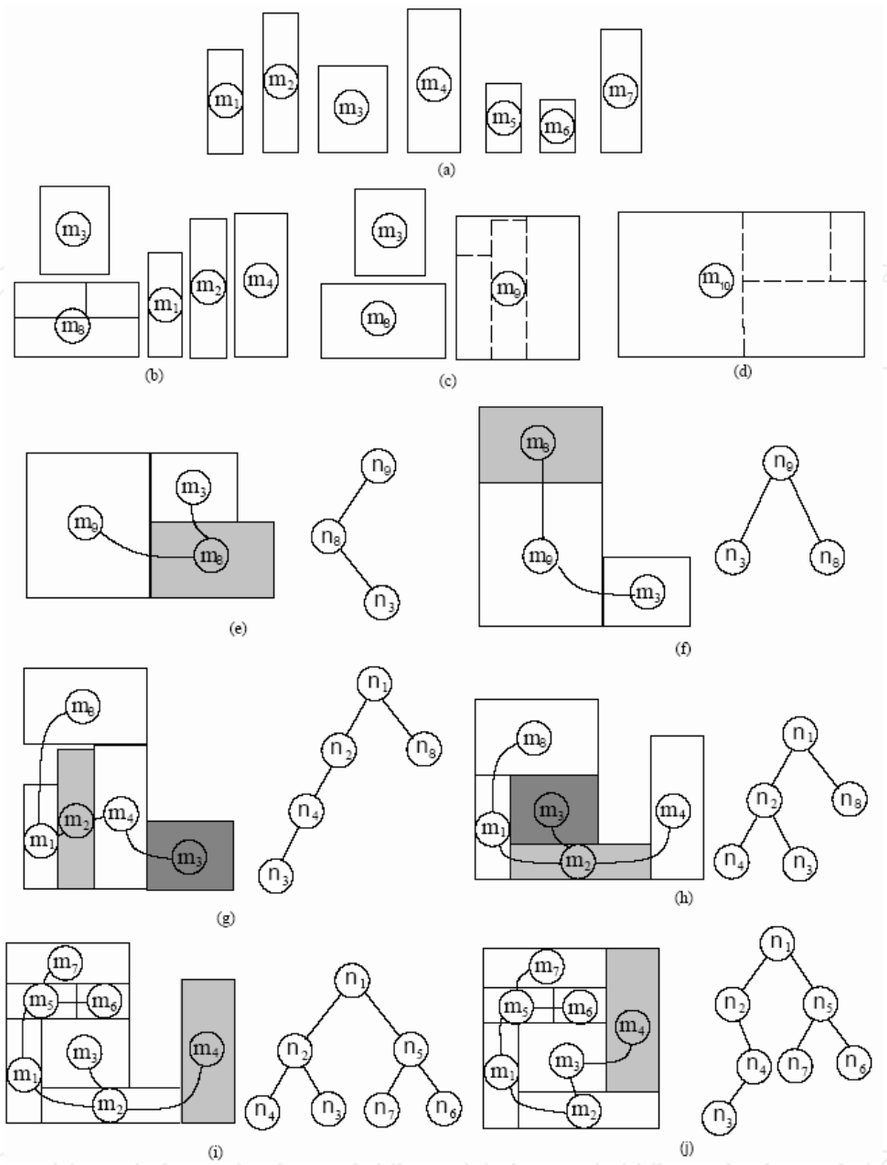


Fig. 11. An example of MBNE algorithm (Lee et al., 2003). In (f), we perform ϵ-neighborhood and λ-exchange refinement.

5. Conclusion

In this work, we have shown improved approaches on the multilevel hierarchical floorplan/placement for large-scale circuits. Our MBNE algorithm uses the improved format of ϵ-neighborhood and λ-exchange algorithm in simulated annealing based multilevel floorplanner. Experimental results have shown that the MBNE algorithm has better performance compared with the MB*-tree, state of the art floorplanner, in several benchmarks.

Package	Area optimization			WL optimization	
	Area(mm^2)	Dead space(%)	Time(min)	WL(mm)	Time(min)
MBNE	671.32/674.57	1.99/2.45	4.00/3.47	53723/58585	150.28/150.18
MB*-tree	673.60/679.41	2.32/3.15	3.95/3.84	55971/59759	180.45/184.54

Package	Simultaneous area and WL optimization			
	Area(mm^2)	Dead space(%)	WL(mm)	Time(min)
MBNE	730.70/742.07	9.95/11.30	63583/63956	150.12/150.10
MB*-tree	769.10/797.28	14.45/17.37	67179/66407	153.96/159.19

Table 1. Comparisons for area optimization alone, wirelength optimization alone, and simultaneous area and wirelength optimization between MBNE and MB*-tree based on the circuit *industry*.

Acknowledgement
We thank Mr. H.-C. Lee and Prof. Y.-W. Chang at National Taiwan University for their MB*-tree platform and benchmarks.

Circuit	# modules	Total area (mm^2)	MB*-tree			MBNE		
			Area (mm^2)	Dead space (%)	Time (min)	Area (mm^2)	Dead space (%)	Time (min)
ami49	49	35.445	36.46	2.79	1.19	36.22	2.14	1.00
ami49_4	196	141.780	146.86	3.46	6.29	144.86	2.12	5.00
ami49_20	980	708.908	732.19	3.18	10.21	727.81	2.60	10.08
ami49_60	2940	2126.724	2211.75	3.84	16.73	2195.76	3.14	15.17
ami49_100	4900	3544.540	3704.65	4.32	20.47	3681.56	3.72	20.18
ami49_150	7350	5316.750	5590.95	4.90	26.77	5560.33	4.38	25.58
ami49_200	9800	7089.808	7478.55	5.21	31.65	7454.86	4.91	30.13

Table 2. Comparisons for area and runtime between MBNE and MB*-tree in fabricated benchmarks from

00	Area optimization			WL optimization	
	Area($0.001mm^2$)	Dead space(%)	Time(min)	WL(mm)	Time(min)
NE	182.490	1.64	5.00	110.982	10.03
tree	184.338	2.62	5.17	111.819	10.89

00	Area optimization			WL optimization	
	Area($0.001mm^2$)	Dead space(%)	Time(min)	WL(mm)	Time(min)
NE	179.452	2.09	7.00	241.696	15.37
tree	180.000	2.39	7.78	244.233	15.94

00	Area optimization			WL optimization	
	Area($0.001mm^2$)	Dead space(%)	Time(min)	WL(mm)	Time(min)
NE	278.964	2.08	10.01	388.162	20.40
tree	279.310	2.20	10.17	391.651	21.45

a and wirelength optimization between MBNE and MB*-tree with GSRC benchmarks.

Circuit	# modules	Total area ($0.001mm^2$)	MB*-tree			MBNE		
			Area ($0.001mm^2$)	Dead space (%)	Time (min)	Area ($0.001mm^2$)	Dead space (%)	Time (min)
industry	189	657,984	673,600	2.32	3.95	673,731	2.34	2.77
n100	100	179.500	184.338	2.62	5.17	183.365	2.11	3.61
ami49_20	980	708,908	732,190	3.18	10.21	729,982	2.89	7.57
ami49_60	2940	2,126,724	2,211,750	3.84	16.73	2,199,793	3.32	11.73

Table 4. Comparisons for efficiency between MBNE and MB*-tree with four benchmarks.

6. References

Alpert, C. J., Huang, J.-H. & Kahng, A. B. (1998). "Multilevel circuit partitioning", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **17**(8): 655–667.

Chan, T. F., Cong, J., Kong, T. & Shinnerl, J. R. (2000). "Multilevel optimization for large-scale circuit placement", *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pp. 171–176.

Chang, Y.-C., Chang, Y.-W., Wu, G.-M. & Wu, S.-W. (2000). "B*-trees: A new representation for non-slicing floorplans", *Proceedings IEEE/ACM Design Automation Conference*, pp. 458–463.

Cong, J., Fang, J. & Zhang, Y. (2001). "Multilevel approach to full-chip gridless routing", *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pp. 396–403.

Cong, J., Xie, M. & Zhang, Y. (2002). "An enhanced multilevel routing system", *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pp. 51–58.

Goto, S. (1981). "An efficient algorithm for the two-dimensional placement problem in electrical circuit layout", *IEEE Transactions on Circuits and Systems* **28**(1): 12–18.

Guo, P.-N., Cheng, C.-K. & Yoshimura, T. (1999). "An O-tree representation of non-slicing floorplan and its applications", *Proceedings IEEE/ACM Design Automation Conference*, pp. 268–273.

Hendrickson, B. & Leland, R. (1995). "A multilevel algorithm for partitioning graph", *Proceedings of Supercomputing*.

Karypis, G. & Kumar, V. (1999). "Multilevel k-way hypergraph partitioning", *Proceedings IEEE/ACM Design Automation Conference*, pp. 343–348.

Lee, H.-C., Chang, Y.-W., Hsu, J.-M. & Yang, H. H. (2003). "Multilevel floorplanning/placement for large-scale modules using B*-trees", *Proceedings IEEE/ACM Design Automation Conference*, pp. 812–817.

Lin, J.-M. & Chang, Y.-W. (2001). "TCG: A transitive closuer graph based representation for non-slicing floorplans", *Proceedings IEEE/ACM Design Automation Conference*, pp. 764–769.

Lin, J.-M. & Chang, Y.-W. (2002a). "TCG-S:Orthogonal coupling of P*-admissible representations for general floorplans", *Proceedings IEEE/ACM Design Automation Conference*, pp. 842–847.

Lin, J.-M., Chang, Y.-W. & Lin, S.-P. (2003). "Corner sequence: A P-admissible floorplan representation with a worst-case linear-time packing scheme", *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **11**(4): 679–686.

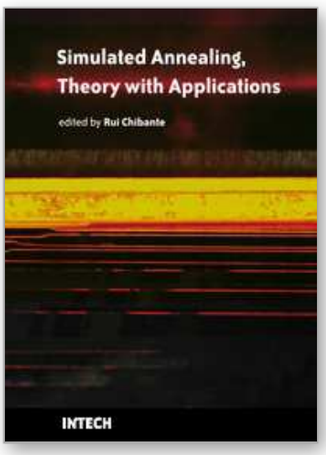
Lin, S.-P. & Chang, Y.-W. (2002b). "A novel framework for multilevel routing considering routability and performance", *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pp. 44–50.

Murata, H., Fujiyoshi, K., Nakatake, S. & Kajitani, Y. (1995). "Rectangle-packing based module placement", *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pp. 472–479.

Nakatake, S., Fujiyoshi, K., Murata, H. & Kajitani, Y. (1996). "Module placement on BSG-structure and IC layout applications", *Proceedings IEEE/ACM International Conference on Computer-Aided Design*, pp. 484–491.

Otten, R. H. J. M. (1982). "Automatic floorplan design", *Proceedings IEEE/ACM Design Automation Conference*, pp. 261–267.

Wong, D. F. & Liu, C. L. (1986). "A new algorithm for floorplan design", *Proceedings IEEE/ACM Design Automation Conference*, pp. 101–107.



Simulated Annealing, Theory with Applications
Edited by Rui Chibante

ISBN 978-953-307-134-3
Hard cover, 292 pages
Publisher Sciyo
Published online 18, August, 2010
Published in print edition August, 2010

The book contains 15 chapters presenting recent contributions of top researchers working with Simulated Annealing (SA). Although it represents a small sample of the research activity on SA, the book will certainly serve as a valuable tool for researchers interested in getting involved in this multidisciplinary field. In fact, one of the salient features is that the book is highly multidisciplinary in terms of application areas since it assembles experts from the fields of Biology, Telecommunications, Geology, Electronics and Medicine.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen